CSE211Digital Design

Akdeniz University

Week6: Boolean Algebra and Logic Gates

Assoc.Prof.Dr. Taner Danışman

tdanisman@akdeniz.edu.tr

Course program

Week 01	2-Oct-23	Introduction
Week 02	9-Oct-23	Digital Systems and Binary Numbers I
Week 03	16-Oct-23	Digital Systems and Binary Numbers II
Week 04	23-Oct-23	Boolean Algebra and Logic Gates I
Week 05	30-Oct-23	Boolean Algebra and Logic Gates II
Week 06	6-Nov-23	Gate Level Minimization
Week 07	13-Nov-23	Karnaugh Maps
Week 08	20-Nov-23	Midterm
Week 09	27-Nov-23	Karnaugh Maps
Week 10	4-Dec-23	Combinational Logic
Week 11	11-Dec-23	Combinational Logic
Week 12	18-Dec-23	Timing, delays and hazards
Week 13	25-Dec-23	Synchronous Sequential Logic
Week 14	1-Jan-24	Synchronous Sequential Logic

New chapter terms to know Reading (Chapter 2.9 Integrated Circuits)

- *IC (Integrated Circuit)*
- *SSI (Small Scale Integration)*
- *MSI (Medium Scale Integration)*
- *LSI (Large Scale Integration)*
- *VLSI (Very Large Scale Integration)*
- *TTL Transistor–Transistor logic;*
- *ECL Emitter-Coupled Logic;*
- *MOS Metal-Oxide Semiconductor;*
- *CMOS Complementary Metal-Oxide Semiconductor*
- *CAD (Computer Aided Design)*

New chapter terms to know Reading (Chapter 2.9 Integrated Circuits)

- *FPGA (Field Programmable Gate Array)*
- *PLD (Programmable Logic Device)*
- *IEEE (Institute of Electronics and Electrical Engineers)*

New chapter terms to know Reading (Chapter 2.9 Integrated Circuits)

- *Fan-out*
- *Fan-in*
- *Power dissipation*
- *Propagation delay*
- *Noise margin*

Primitive Data Types with size modifiers

Data type	Size	Value range
char	1	-128 to 127 or 0 to 255
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2 **or** 4	-32,768 to 32,767 **or** -2,147,483,648 to 2,147,483,647
unsigned int	2 **or** 4	0 to 65,535 **or** 0 to 4,294,967,295
short	2	-32,768 to 32,767
unsigned short	2	0 to 65,535
long	4	-2,147,483,648 to 2,147,483,647
unsigned long	4	0 to 4,294,967,295

Boolean Algebra – DeMorgan's Law

- Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly.
- DeMorgan's law provides an easy way of finding the complement of a Boolean function.
- Recall **DeMorgan's law** states:

$$\overline{(xy)} = \bar{x} + \bar{y} \quad \text{and} \quad \overline{(x+y)} = \bar{x}\bar{y}$$

Boolean Algebra – Complement of Functions

EXAMPLE 2.2

Find the complement of the functions $F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$. By applying DeMorgan's theorems as many times as necessary, the complements are obtained as follows:

$$F_1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z')$$

$$F_2' = [x(y'z' + yz)]' = x' + (y'z' + yz)' = x' + (y'z')'(yz)'$$

$$= x' + (y + z)(y' + z')$$

$$= x' + yz' + y'z$$

Boolean Algebra – Duality

- **Examples:**

The dual of x(y + z) is **x + yz**.

The dual of x·1 + (y + z) is **(x + 0)(yz)**.

The **dual** of a Boolean function F represented by a Boolean expression is the function represented by the dual of this expression.

This dual function, denoted by Fd, does not depend on the particular Boolean expression used to represent F.

Boolean Algebra – Minterm Canonical Formula

X	Y	Z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$\overline{x}\,\overline{y}\,z$

$\overline{x}\,y\,z$

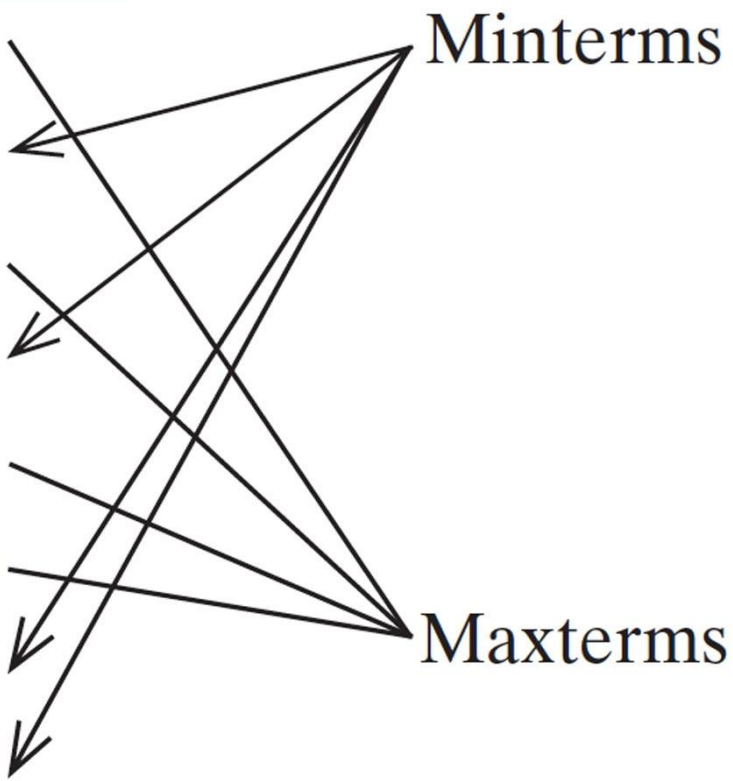
$x\,\overline{y}\,\overline{z}$

$$f(x,y,z) = \overline{x}\,\overline{y}\,z + \overline{x}y\,z + x\,\overline{y}\,\overline{z}$$

Table 2.6

Truth Table for F = xy + x′z

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Boolean Algebra – Maxterm Canonical Formula

X	Y	Z	f	
0	0	0	0	$x + y + z$
0	0	1	1	
0	1	0	0	$x + \overline{y} + z$
0	1	1	1	
1	0	0	1	$\overline{x} + y + \overline{z}$
1	0	1	0	
1	1	0	0	$\overline{x} + \overline{y} + z$
1	1	1	0	$\overline{x} + \overline{y} + \overline{z}$

$$f(x, y, z) = (x + y + z)(x + \overline{y} + z)$$
$$(\overline{x} + y + \overline{z})(\overline{x} + \overline{y} + z)(\overline{x} + \overline{y} + \overline{z})$$

Boolean Algebra – Sum of Products

- The sum-of-products form for our function is:

$$F(x,y,z) = \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}\bar{z} + xy\bar{z} + xyz$$

We note that this function is not in simplest terms. Our aim is only to rewrite our function in canonical sum-of-products form.

$$F(x,y,z) = x\bar{z} + y$$

x	y	z	$x\bar{z}+y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Table 2.4
Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2.3
Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Minterms

EXAMPLE 2.4

Express the Boolean function $F = A + B'C$ as a sum of minterms. The function has three variables: A, B, and C. The first term A is missing two variables; therefore,

$$A = A(B + B') = AB + AB'$$

This function is still missing one variable, so

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

The second term $B'C$ is missing one variable; hence,

$$B'C = B'C(A + A') = AB'C + A'B'C$$

Combining all terms, we have

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + A'B'C \end{aligned}$$

Minterms (Cont.)

The second term $B'C$ is missing one variable; hence,

$$B'C = B'C(A + A') = AB'C + A'B'C$$

Combining all terms, we have

$$F = A + B'C$$
$$= ABC + ABC' + AB'C + AB'C' + A'B'C$$

But $AB'C$ appears twice, and according to theorem 1 ($x + x = x$), it is possible to remove one of those occurrences. Rearranging the minterms in ascending order, we finally obtain

$$F = A'B'C + AB'C' + AB'C + ABC' + ABC$$
$$= m_1 + m_4 + m_5 + m_6 + m_7$$

When a Boolean function is in its sum-of-minterms form, it is sometimes convenient to express the function in the following brief notation:

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

Maxterms

EXAMPLE 2.5

Express the Boolean function $F = xy + x'z$ as a product of maxterms. First, convert the function into OR terms by using the distributive law:

$$\begin{aligned} F &= xy + x'z = (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

The function has three variables: x, y, and z. Each OR term is missing one variable; therefore,

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

Combining all the terms and removing those which appear more than once, we finally obtain

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0M_2M_4M_5 \end{aligned}$$

A convenient way to express this function is as follows:

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

Conversion between Canonical forms

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. This is because the original function is expressed by those minterms which make the function equal to 1, whereas its complement is a 1 for those minterms for which the function is a 0. As an example, consider the function

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

This function has a complement that can be expressed as

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

Now, if we take the complement of F' by DeMorgan's theorem, we obtain F in a different form:

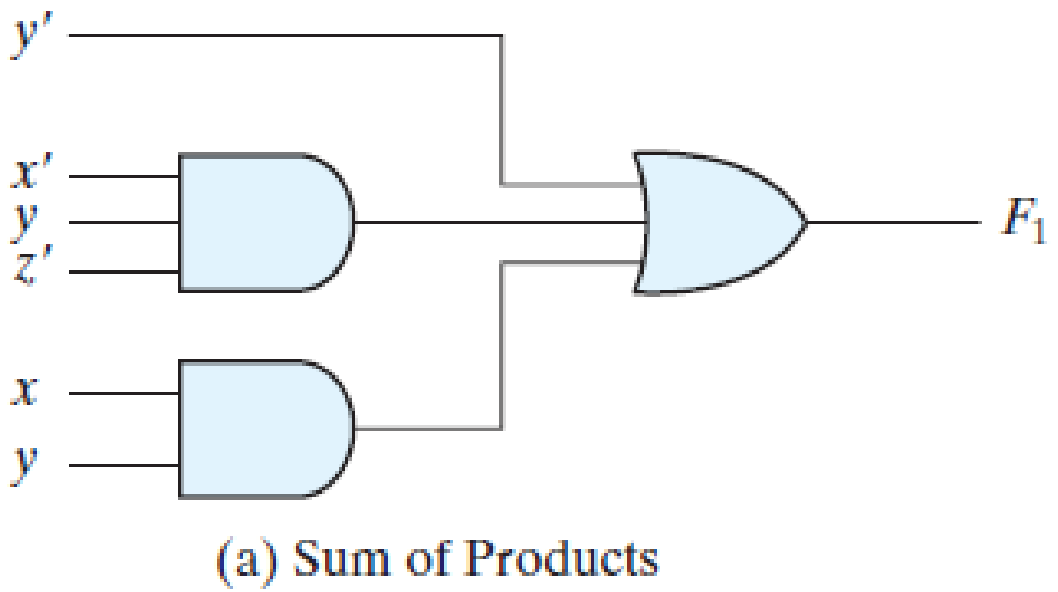
$$F = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

The last conversion follows from the definition of minterms and maxterms as shown in Table 2.3. From the table, it is clear that the following relation holds:

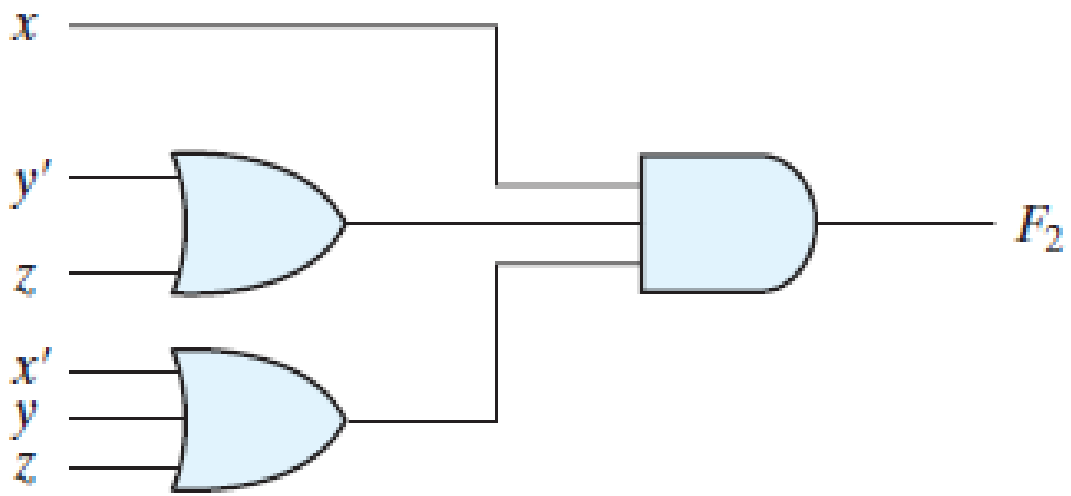
$$m_j' = M_j$$

That is, the **maxterm with subscript j is a complement of the minterm with the same subscript j and vice versa.**

Standard Forms with Logic Gates
Two Level Implementation



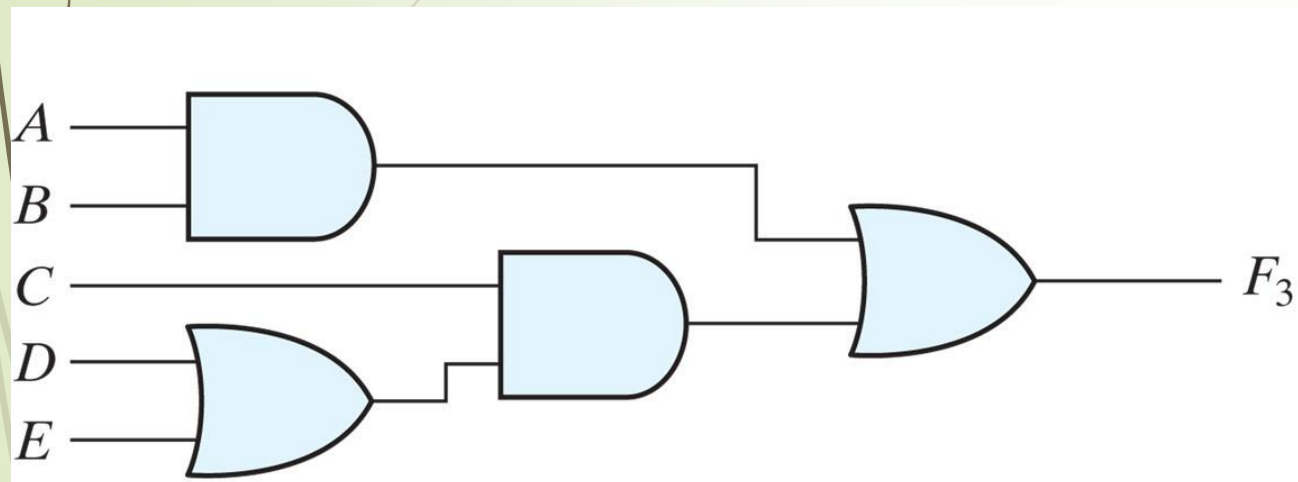
(a) Sum of Products



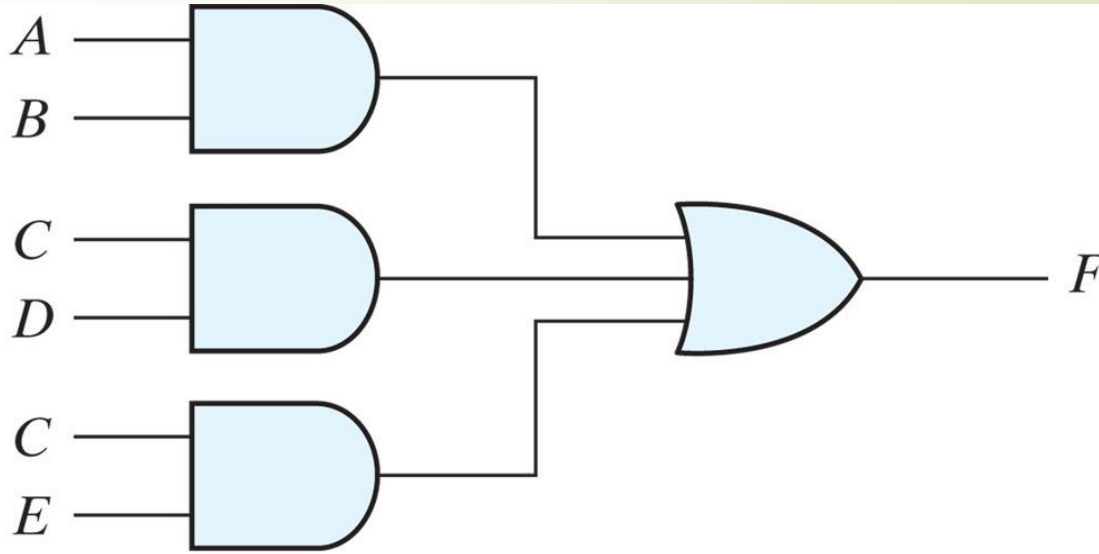
(b) Product of Sums

FIGURE 2.3
Two-level implementation

Three and two level implementation



(a) $AB + C(D + E)$



(b) $AB + CD + CE$

The truth tables for the 16 functions formed with two binary variables

Table 2.7

Truth Tables for the 16 Functions of Two Binary Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

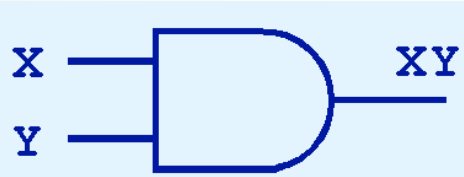
16 Functions of Two variables

Table 2.8
Boolean Expressions for the 16 Functions of Two Variables

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

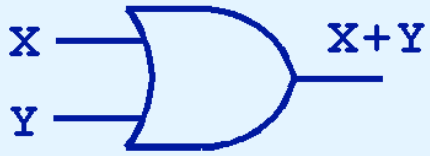
Logic Gates – AND OR NOT

- The three simplest gates are the AND, OR, and NOT gates.



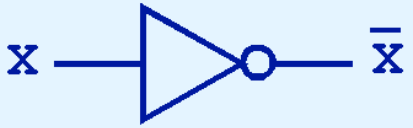
X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1



X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1



NOT X

X	$\overline{X}$
0	1
1	0

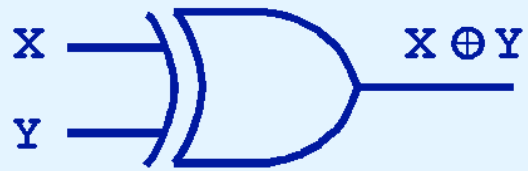
- They correspond directly to their respective Boolean operations, as you can see by their truth tables.

Logic Gates – XOR

- Another very useful gate is the exclusive OR (XOR) gate.
- The output of the XOR operation is true only when the values of the inputs differ.

X XOR Y

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0



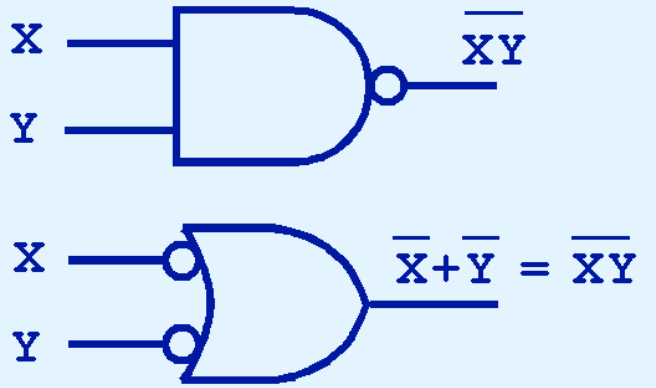
Note the special symbol ⊕ for the XOR operation.

Logic Gates – NAND NOR

- NAND and NOR are two very important gates.
- Their symbols and truth tables are shown at the right.

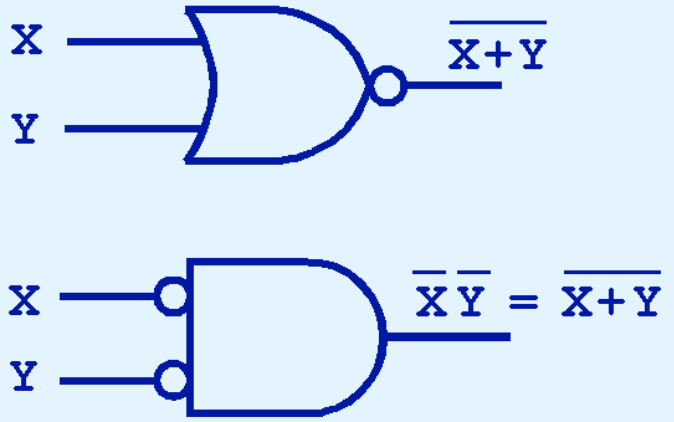
X NAND Y

X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

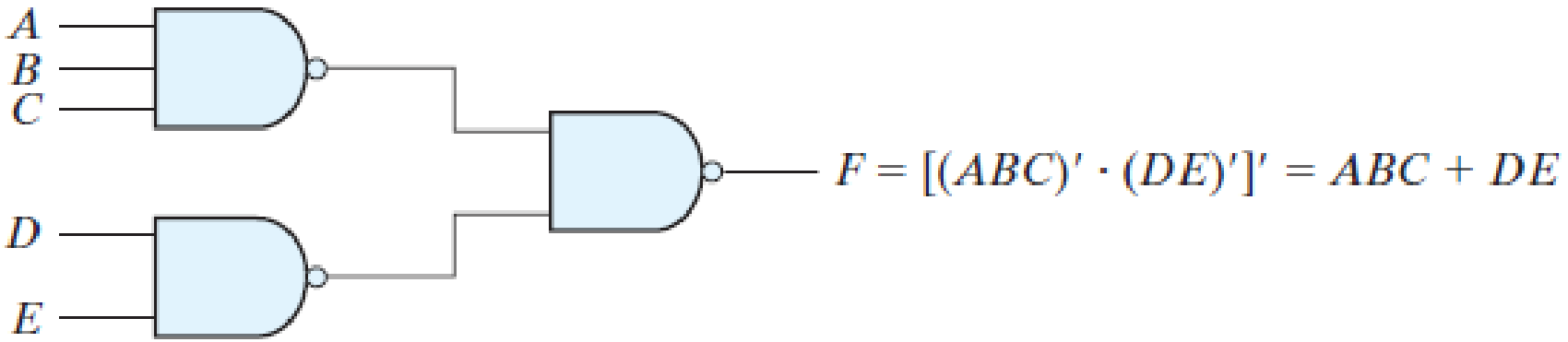


X NOR Y

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0



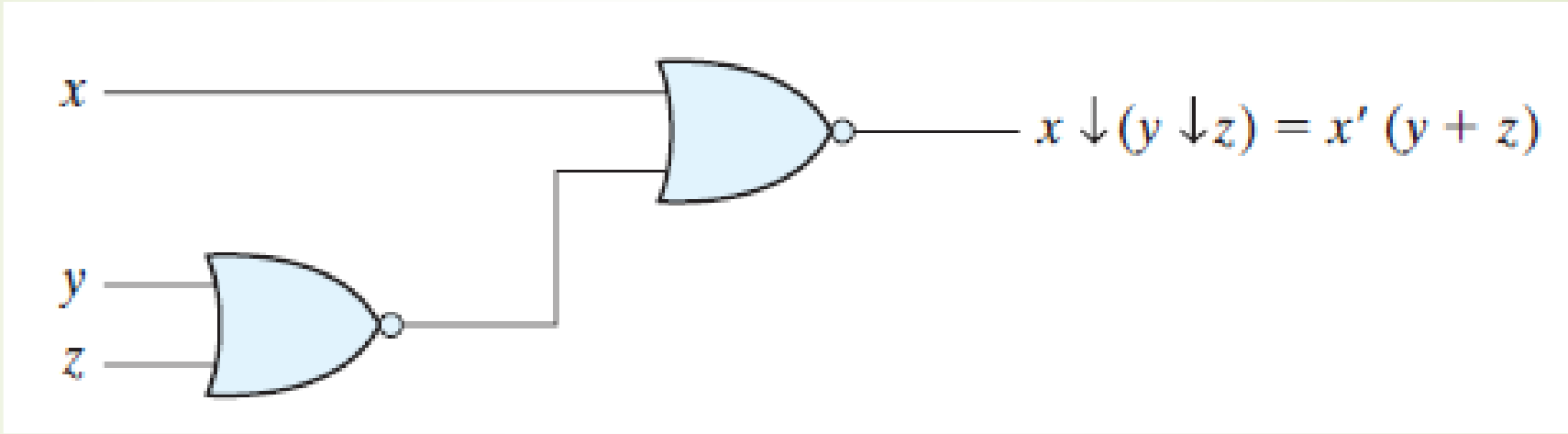
Cascaded NAND gate

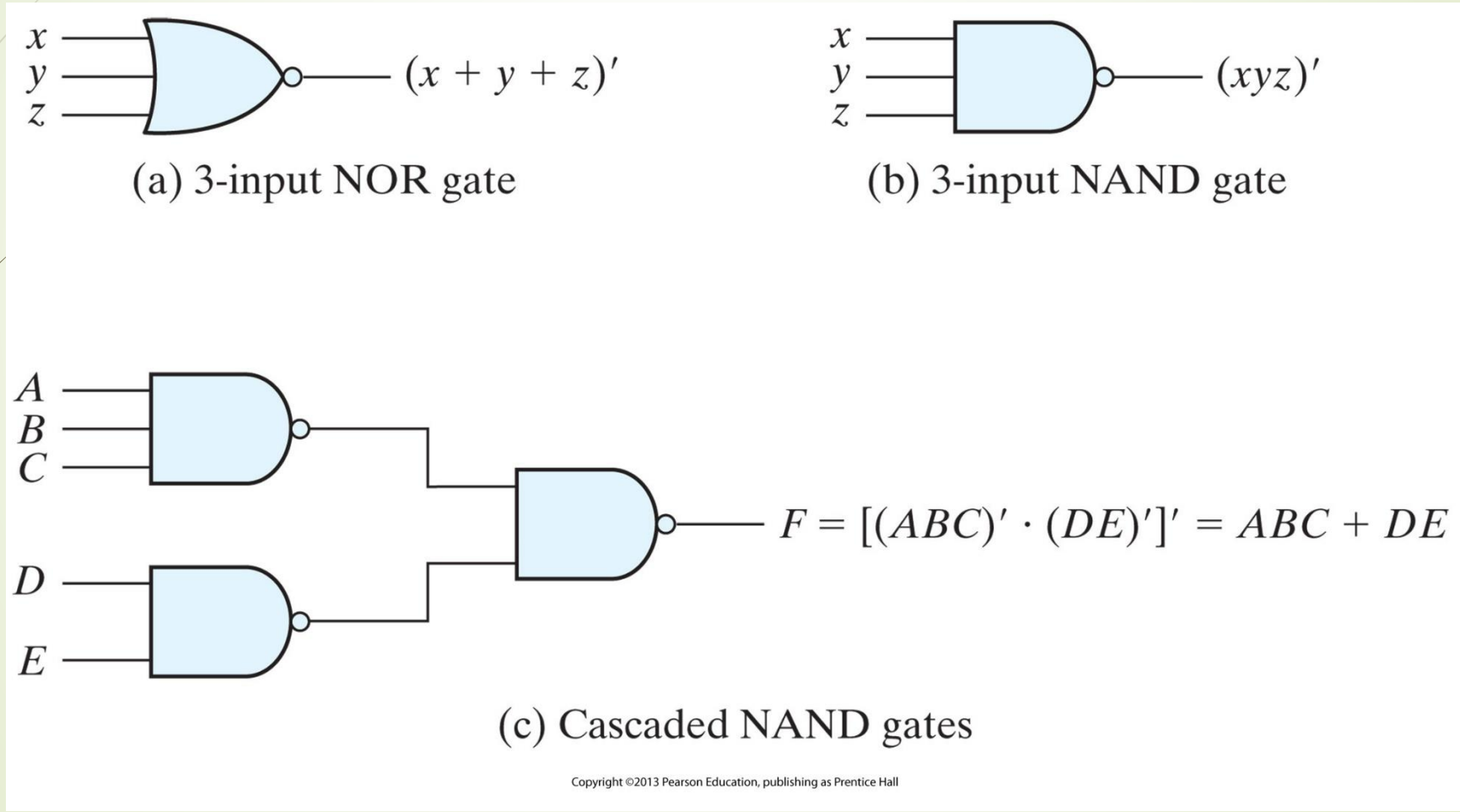


(c) Cascaded NAND gates

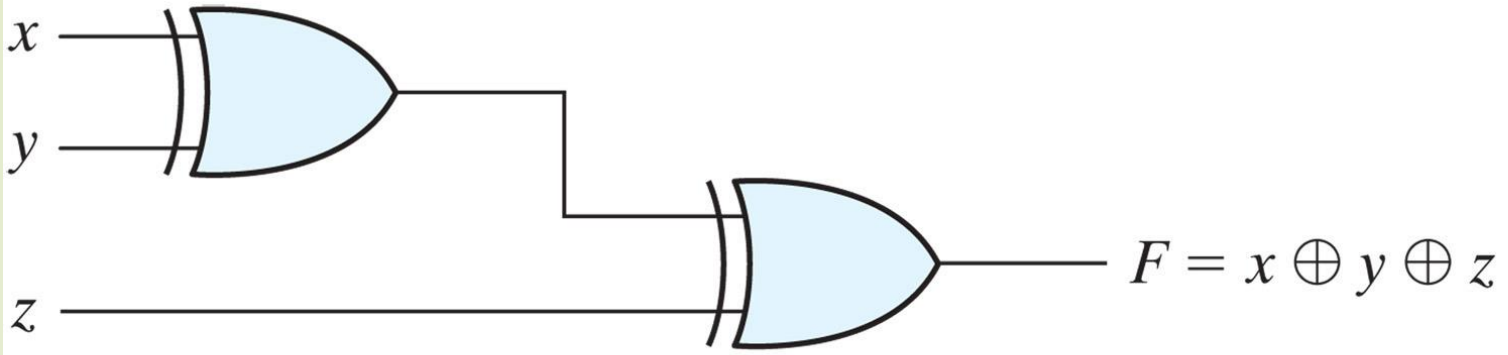
FIGURE 2.7
Multiple-input and cascaded NOR and NAND gates

Cascaded NOR gate





Three input XOR Gate



(a) Using 2-input gates

(b) 3-input gate

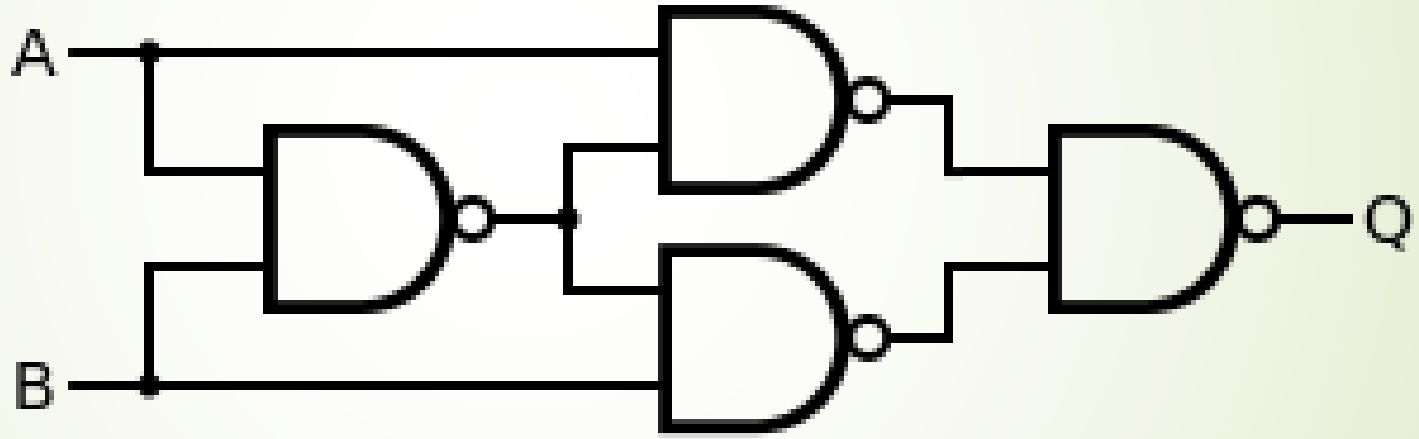
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Truth table

Draw the logic diagram for F=(AB)'+C'D using NAND gate only

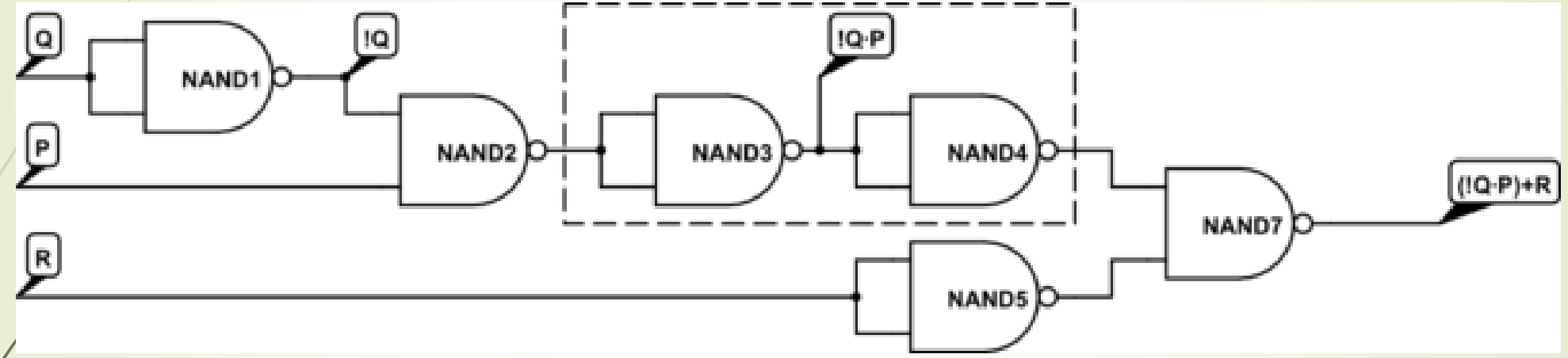
Draw the logic diagram for F=(AB)'+C'D using NOR gate only

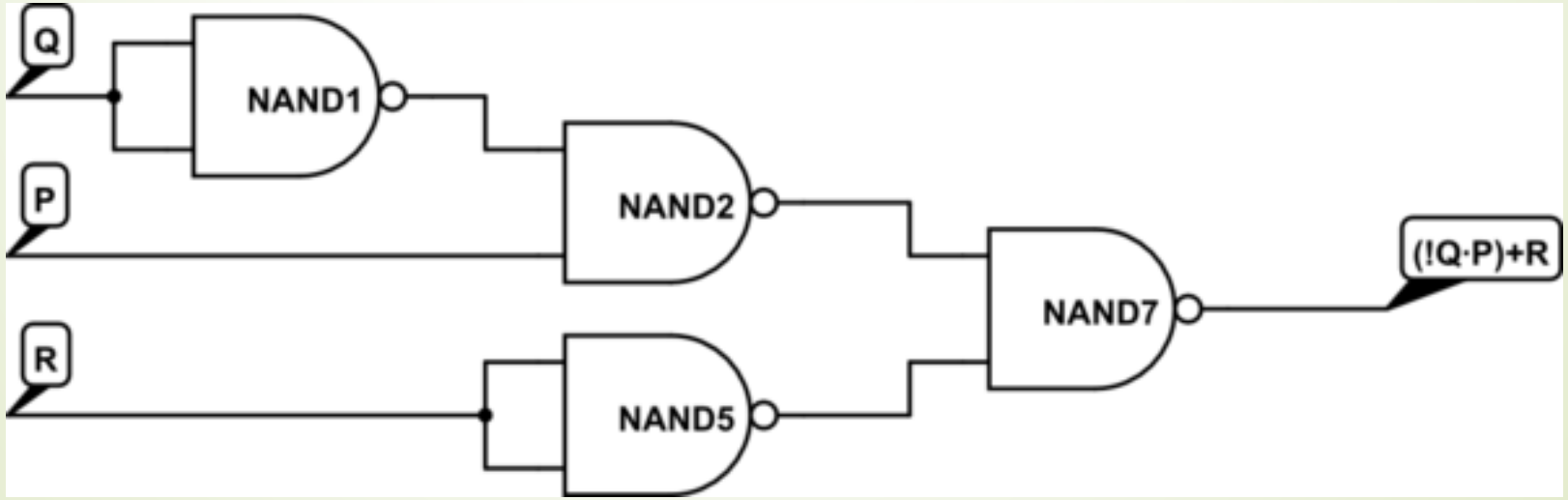
What is the output of this logic diagram?



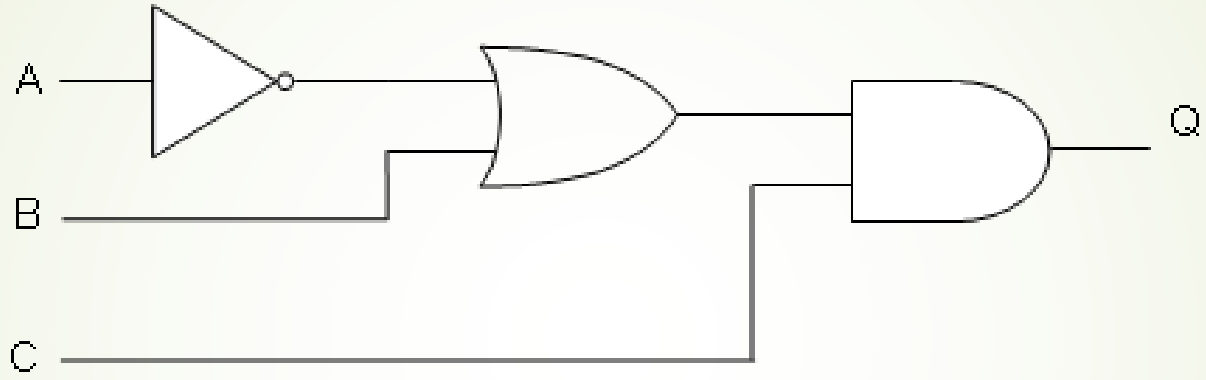
- **Q=A'B+B'A= A XOR B**

Draw the logic diagram for F=(Q'P)+R using NAND gate only

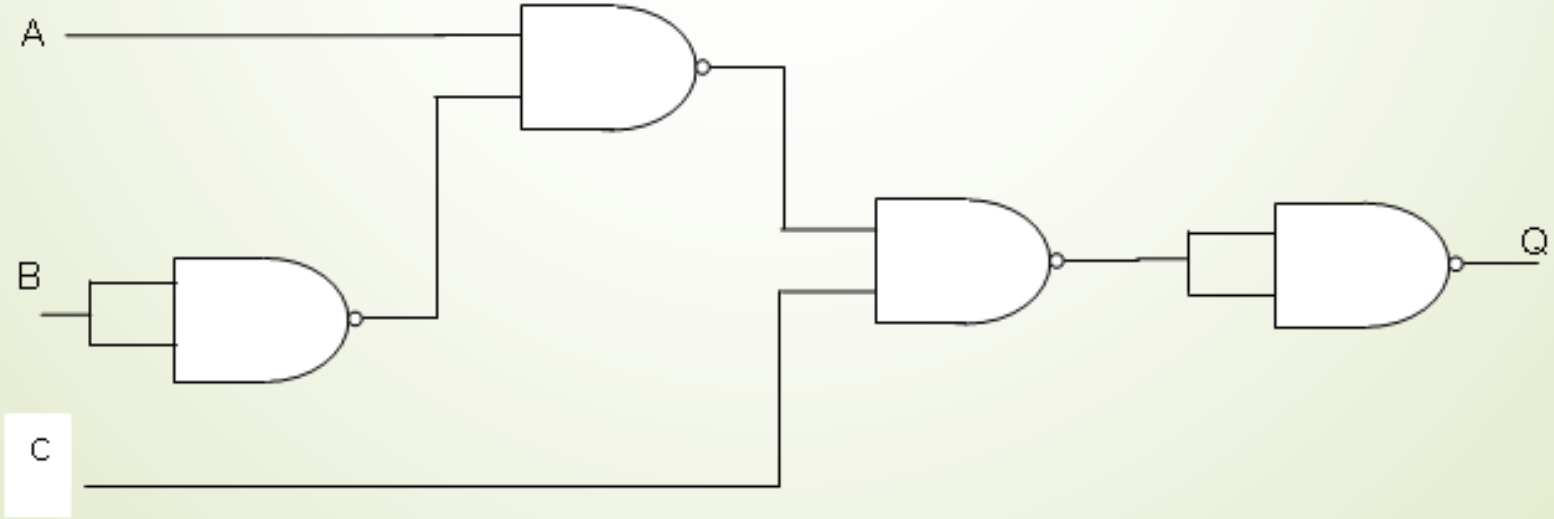




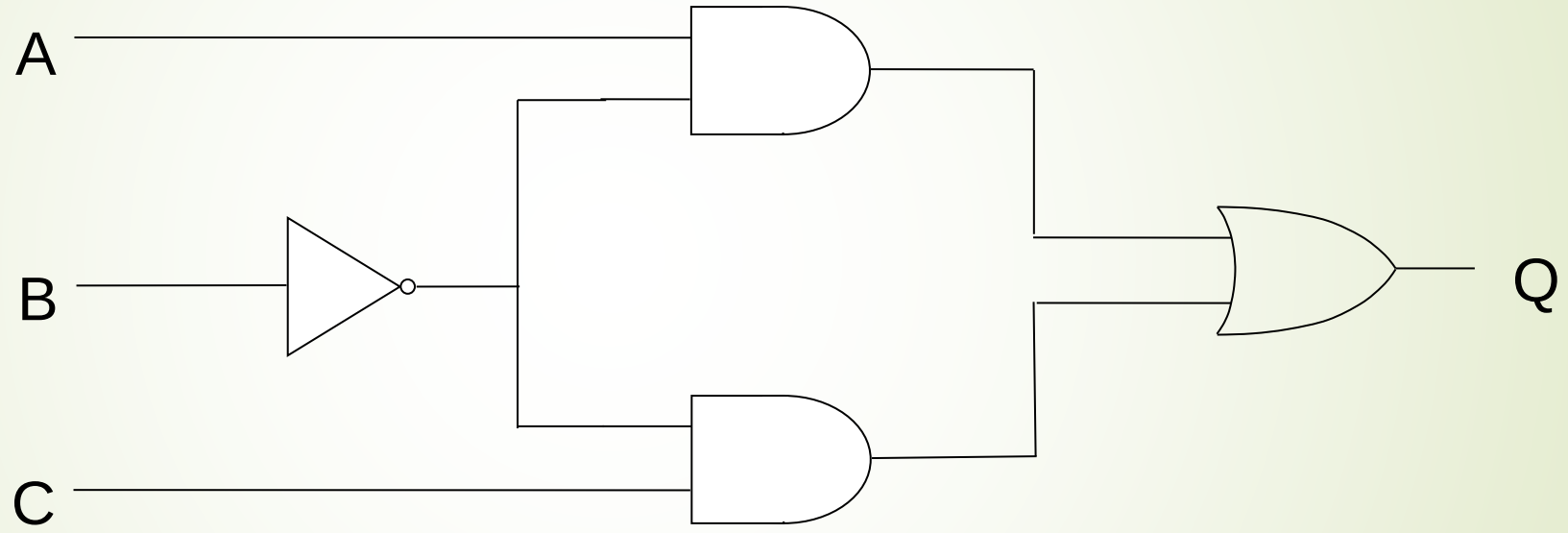
Convert the following logic diagram into NAND gates only (Ex.1)



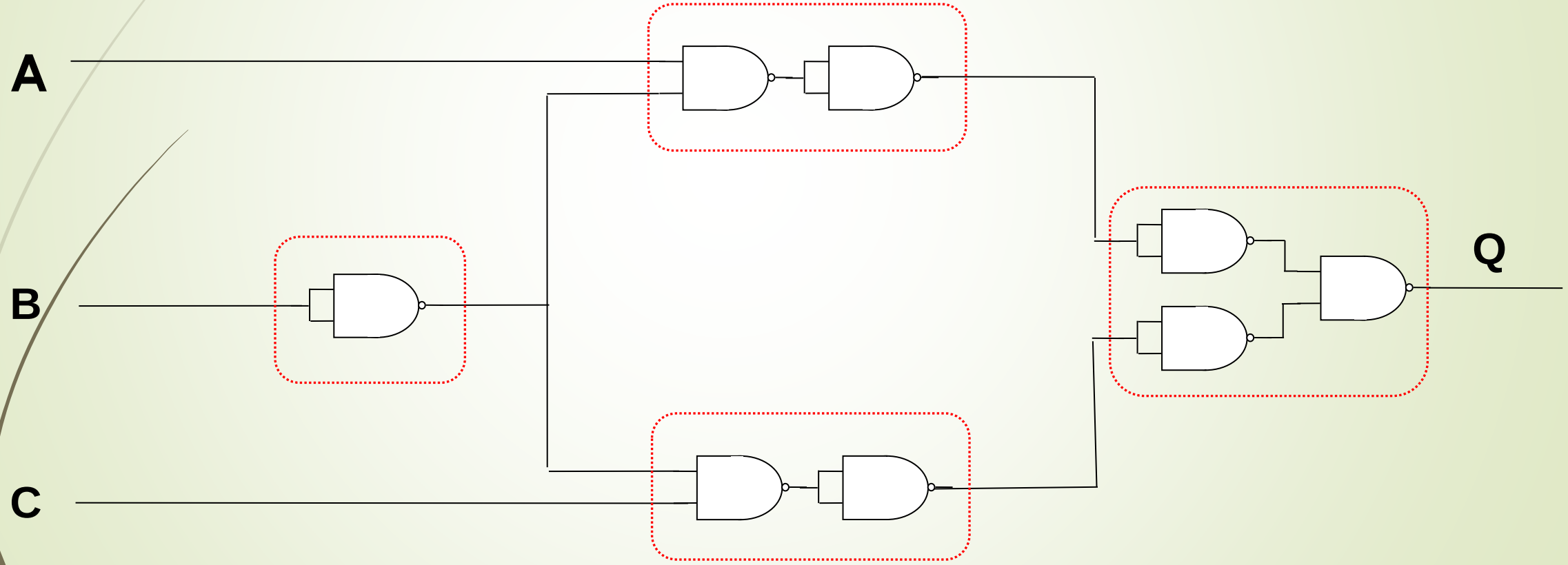
Solution
Q=(A'+B)C



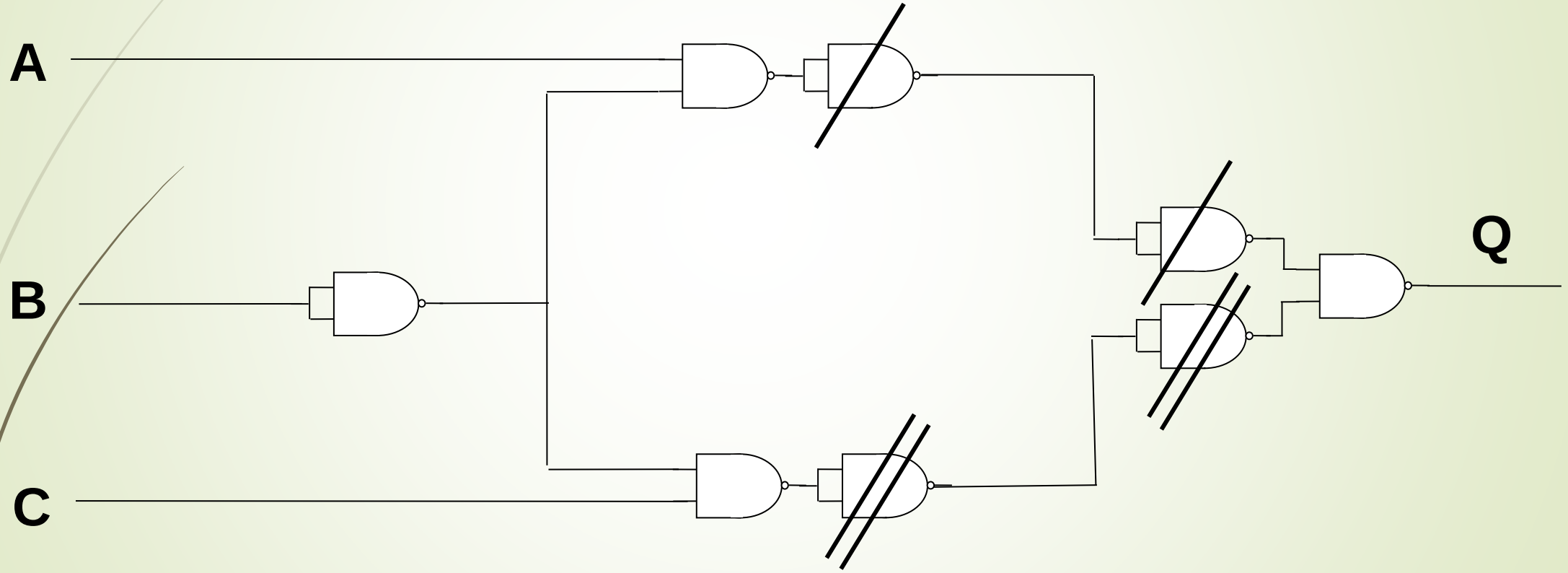
Convert the following logic diagram into NAND gates only (Ex.2)



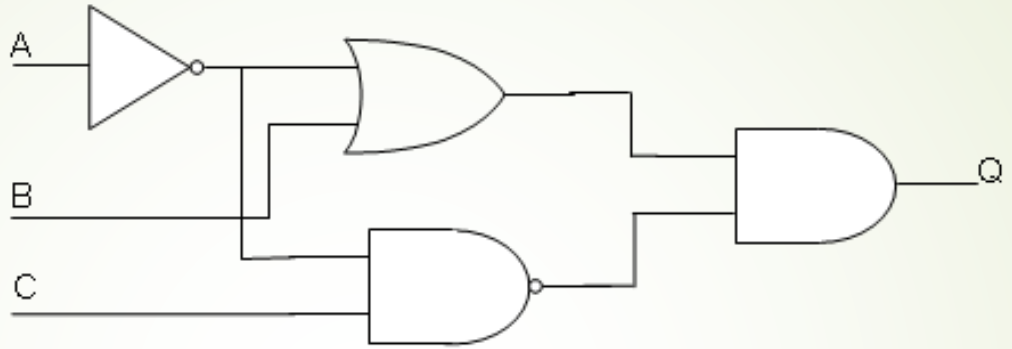
First of all we will replace all of these gates with their NAND equivalent and connect them together

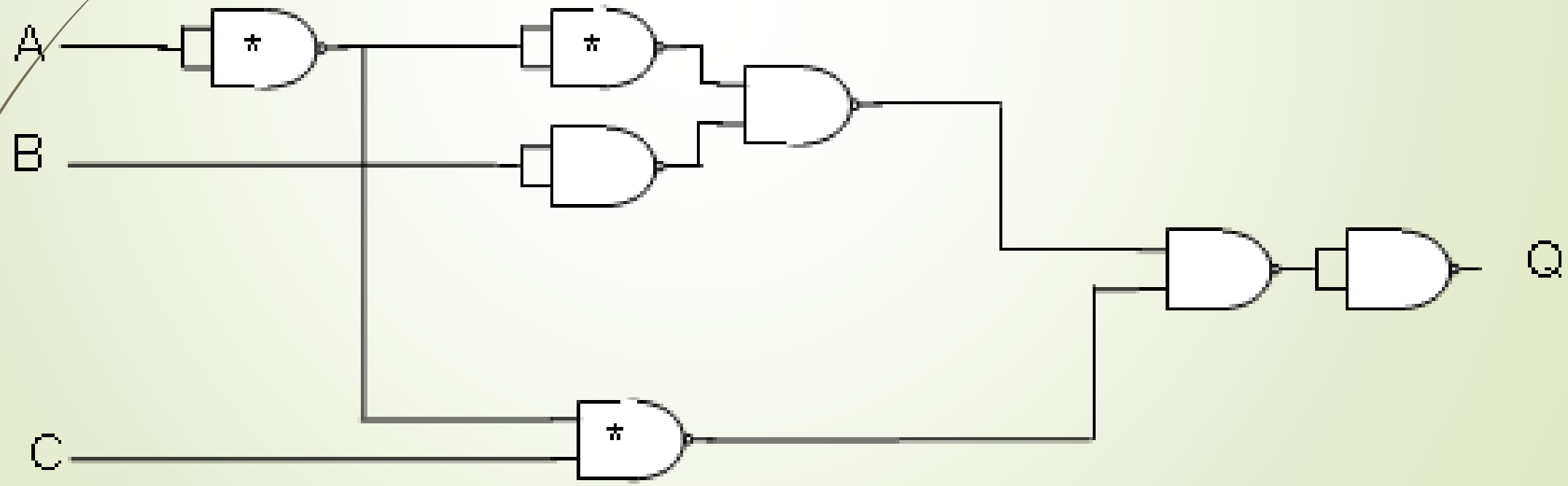


Finally we check for any redundant gates, and identify these.

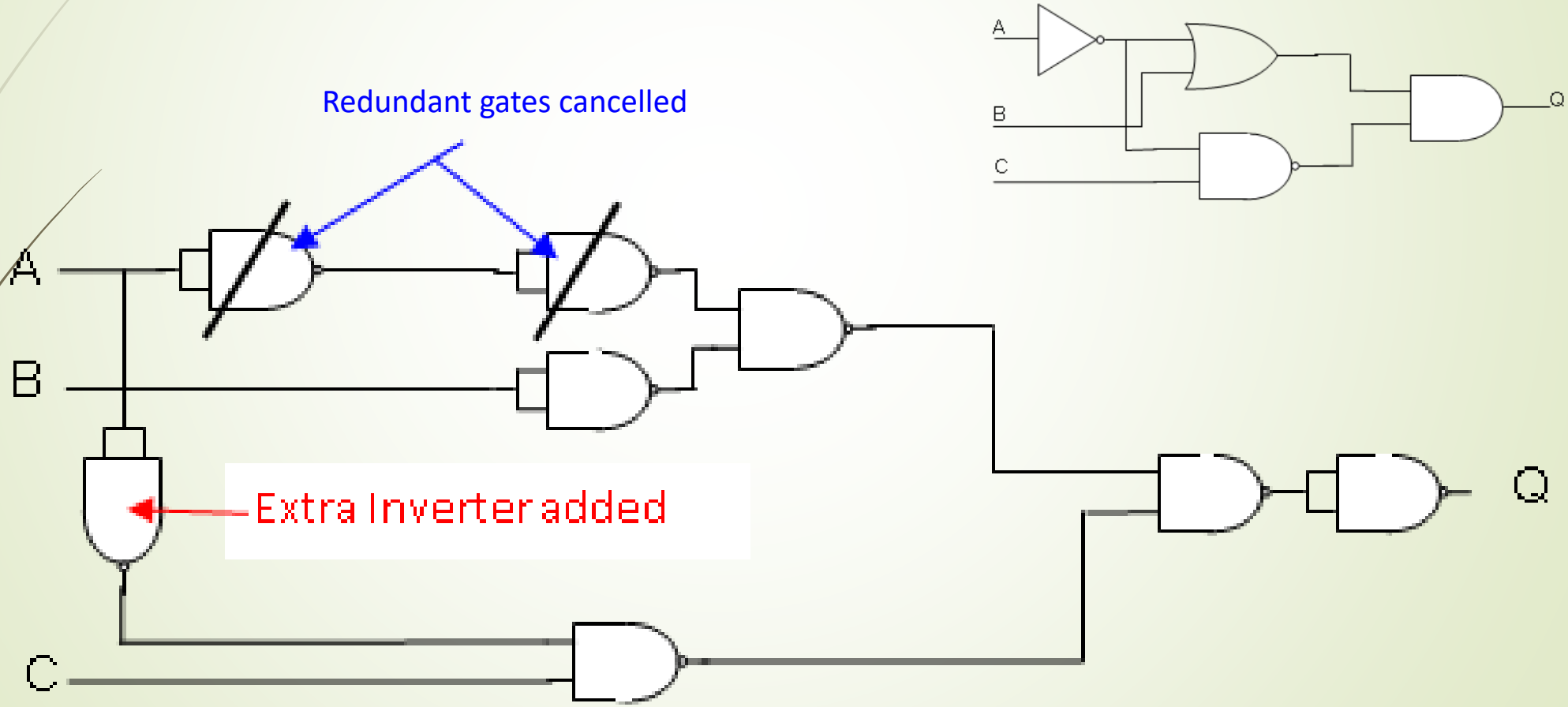


Redundancy in conversions

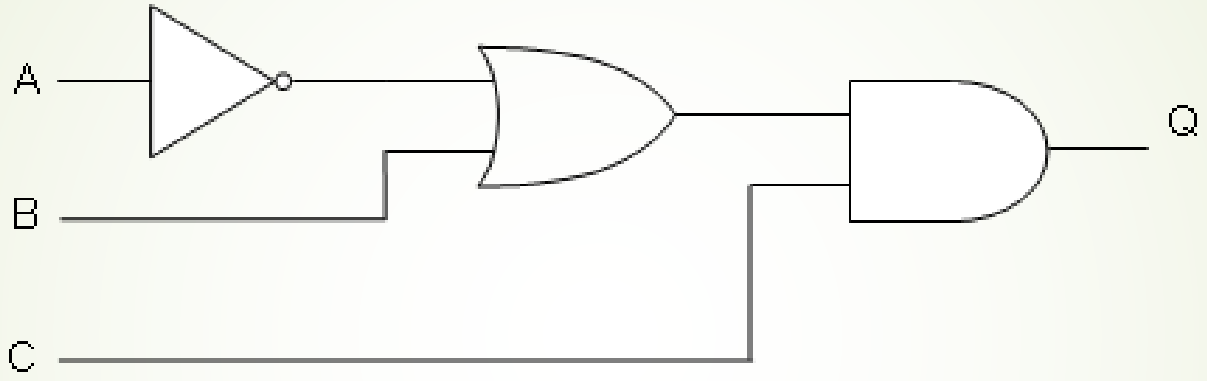




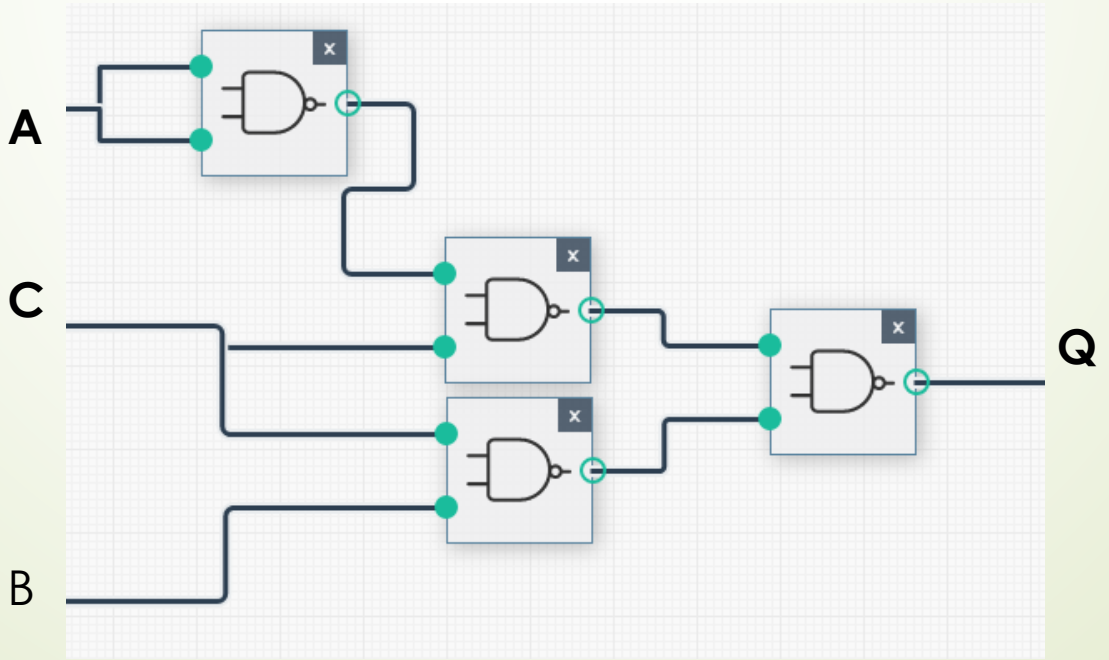
Redundancy in conversions



Create equivalent diagram using NAND gates only (Second Solution)



Solution
Q=(A'+B)C



Show how to create an exclusive-OR gate using only 2-input NAND gates.

Is it possible to write two different truth tables which have the same Boolean expression?

- No. If a Boolean expression satisfies one truth table, it cannot satisfy the other.

Signal Assignment and Logic Polarity